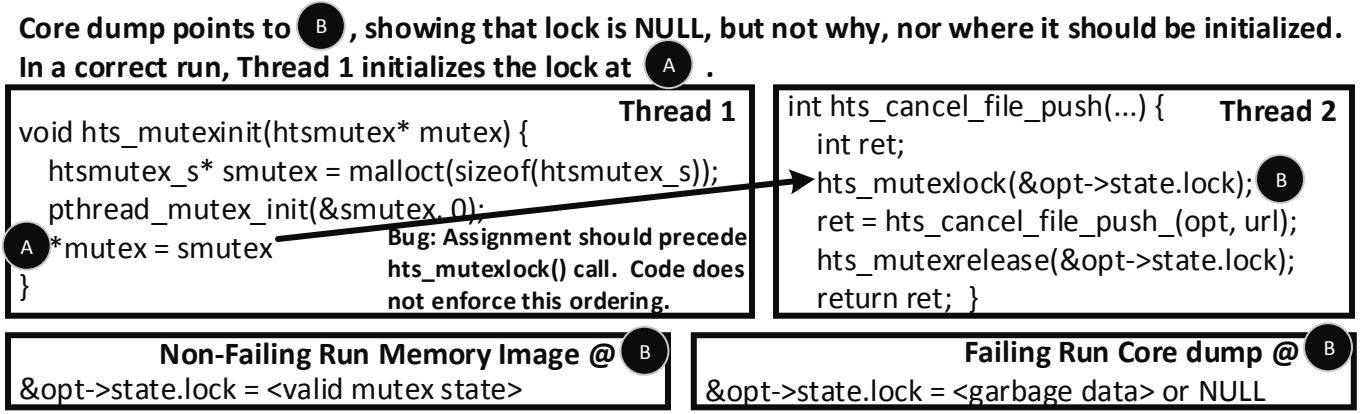
Core dump points to B, showing that lock is NULL, but not why, nor where it should be initialized.
In a correct run, Thread 1 initializes the lock at A.
Thread 1
void hts_mutexinit(htsmutex* mutex) {
htsmutex_s* smutex = malloct(sizeof(htsmutex_s));
pthread_mutex_init(&smutex, 0);
A *mutex = smutex
}
Bug: Assignment should precede hts_mutexlock() call. Code does not enforce this ordering.
Thread 2
int hts_cancel_file_push(...) {
int ret;
hts_mutexlock(&opt->state.lock); B
ret = hts_cancel_file_push_(opt, url);
hts_mutexrelease(&opt->state.lock);
return ret; }
Non-Failing Run Memory Image @ B
&opt->state.lock = <valid mutex state>
Failing Run Core dump @ B
&opt->state.lock = <garbage data> or NULL